

CYSE 420 — Applied Machine Learning in Cybersecurity

Assignment 02: Spam Detection with Logistic Regression and Naive Bayes

Carl Lochstampfor • Fall 2026

1. Introduction

This report compares two supervised classifiers, Logistic Regression and Multinomial Naive Bayes, for the task of labeling SMS messages as spam or ham. The dataset (spam.csv) contains 5,572 messages. Each model was trained and evaluated twice, once with a 70/30 train/test split and once with a 90/10 split, so that the effect of training-set size could be examined alongside the choice of model. Evaluation used accuracy, precision, recall, F1, and ROC AUC, with confusion matrices and ROC curves as visual support.

2. Dataset Preprocessing

Inspection of the raw file showed three problems that had to be resolved before modeling:

- **Junk columns.** The file has five columns, but only v1 (label) and v2 (message) carry data. The three "Unnamed" columns are empty in more than 99% of rows and were dropped. The two remaining columns were renamed to label and message.
- **Duplicate rows.** 403 messages appear more than once. They were removed. If a duplicated message lands in both the training and test set, the model is scored on something it has already memorized, which inflates the test results. Removing duplicates costs 7% of the data but keeps the 70/30 versus 90/10 comparison honest and more compelling than with the inflated percentage. After removing the duplicates, the dataset has a total of 5,169 messages.
- **Class imbalance.** The cleaned data contains 4,516 ham and 653 spam messages, so spam is 12.6% of the total. This imbalance shapes the whole evaluation (Section 5) and is the reason accuracy is not the headline metric.

Two further steps prepared the data for scikit-learn. First, the text labels were label-encoded as ham = 0 and spam = 1 (module 2.2.3), which makes spam the positive class for precision and recall. Second, the messages were converted to numeric features with a bag-of-words representation using CountVectorizer, with lowercasing and English stop-word removal as described in module 2.7.8. Feature scaling (module 2.3) was not applied: the result is a sparse matrix of word counts and standardizing it would destroy the sparsity that both models rely on.

The train/test split was performed before vectorization, and the vectorizer was fit on the training messages only. The test messages were then transformed with the training vocabulary. Fitting on the full dataset would leak test-set words into the vocabulary and overstate performance. Both splits used stratify=y so that each partition kept the 12.6% spam ratio, and random_state=42 so the results are reproducible. The 70/30 split produced 3,618 training and 1,551 test messages with a vocabulary of 6,756 words; the 90/10 split produced 4,652 training and 517 test messages.

3. Models

Logistic Regression. scikit-learn's LogisticRegression with max_iter=1000 (module 2.7.9). The model fits a weight for every word in the vocabulary and passes the weighted sum through the sigmoid function to produce a spam probability, which is compared against a 0.5 threshold (module 2.7.3 and 2.7.5).

Naive Bayes. scikit-learn's MultinomialNB. Of the three Naive Bayes variants available, the multinomial form is the one designed for word-count features; it estimates, for each class, the probability of each word occurring, and combines them under the conditional-independence assumption described in module 2.8.1. GaussianNB assumes continuous normally distributed features and is not appropriate for a count matrix.

Both models were trained on the same feature matrices and evaluated on the same held-out messages within each split, so the comparison isolates the choice of model.

4. Results

Table 1 summarizes all four runs. Figure 1 shows the confusion matrices and Figure 2 the ROC curves.

Table 1. Test-set performance for both models on both splits (spam = positive class).

Model / Split	Accuracy	Precision	Recall	F1	ROC AUC
Logistic Regression 70/30	0.9723	0.9935	0.7857	0.8775	0.9886
Naive Bayes 70/30	0.9852	0.9727	0.9082	0.9393	0.9861
Logistic Regression 90/10	0.9787	1.0000	0.8308	0.9076	0.9822
Naive Bayes 90/10	0.9826	0.9516	0.9077	0.9291	0.9865

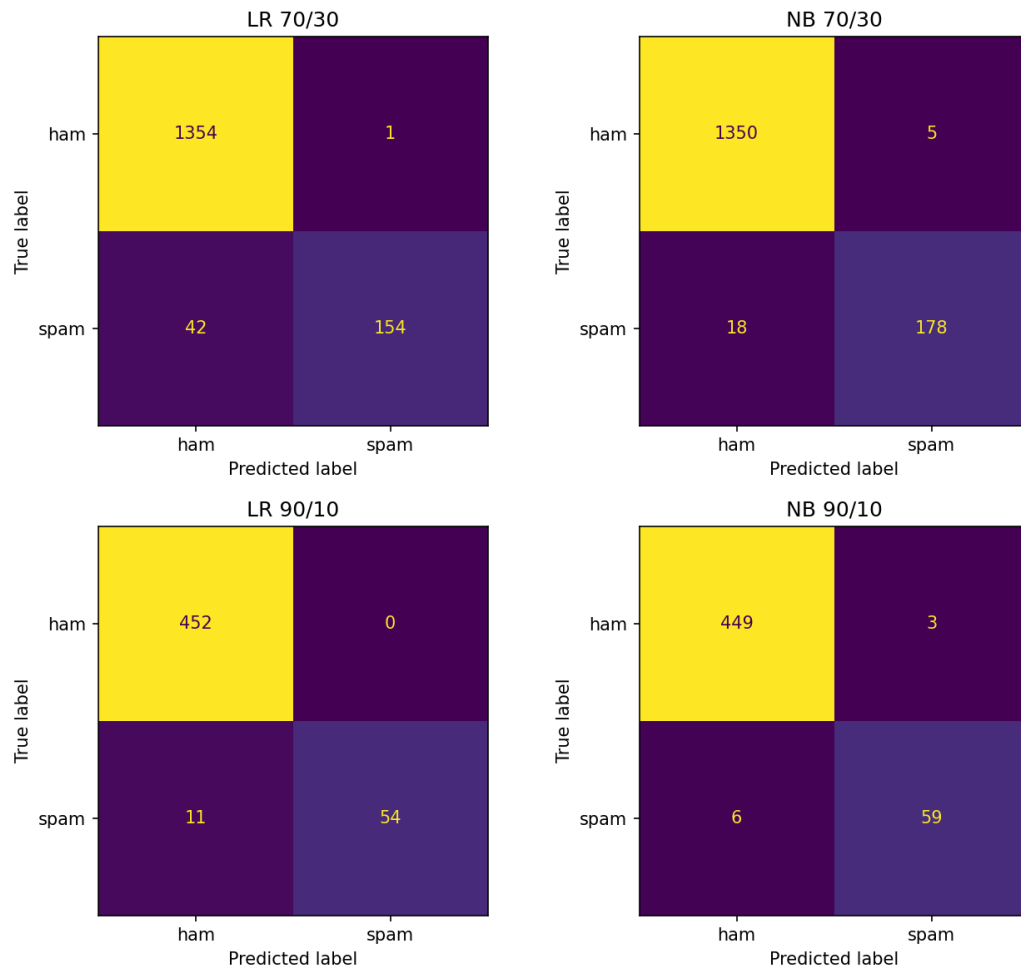


Figure 1. Confusion matrices. Rows are true labels, columns are predicted labels.

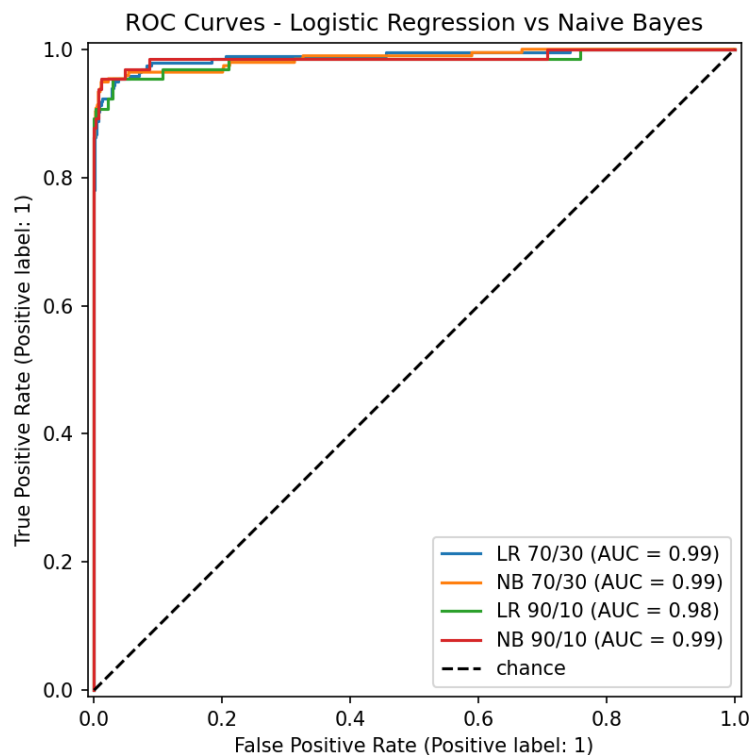


Figure 2. ROC curves for all four model/split combinations.

5. Interpretation and Comparison

Accuracy is misleading on this dataset. With 12.6% spam, a model that labels every message "ham" scores 87.4% accuracy without learning anything. All four models sit between 97.2% and 98.5%, so accuracy barely separates them. Recall and F1 do as well (see module 2.8.5 regarding imbalanced spam data).

Naive Bayes catches more spam; Logistic Regression flags fewer real messages. In the 70/30 split, NB recall is 0.908 versus 0.786 for LR: NB missed 18 spam messages, LR missed 42 (Figure 1). LR's precision is 0.9935 with a single false positive, against 0.973 and five false positives for NB. LR is the conservative model. It rarely mislabels a legitimate message, but it lets spam through. NB has the higher F1 in both splits and is the better balance for this dataset. *Which trade-off is preferable depends on the cost of each error:* module 2.9.2 notes that a false positive means a legitimate message is hidden from the user, while a false negative means spam reaches the inbox.

Why Naive Bayes does well on text. MultinomialNB models word counts directly, and the conditional-independence assumption is a reasonable approximation for bag-of-words features, where the presence of "free" or "winner" is informative on its own. It also needs relatively little data to estimate its per-word probabilities, which is why its scores barely move between the 70/30 and 90/10 splits.

Effect of the split. Logistic Regression improved with more training data: recall rose from 0.786 to 0.831 and precision reached 1.000. Naive Bayes was flat, with recall of 0.908 in both cases. However, the 90/10 test set contains only 65 spam messages, so a single miss moves recall by 1.5 percentage points. LR's improvement amounts to three fewer missed messages and should not be overread. The 70/30 split gives more trustworthy estimates because the test set is three times larger; the 90/10 split gives the model more to learn from (the trade-off described in module 2.4).

ROC AUC is 0.98 to 0.99 for all four runs. AUC is threshold-independent (module 2.7.7), so it measures how well each model ranks spam above ham regardless of the cutoff. On that measure the two models are almost indistinguishable (Figure 2). The recall gap at the default 0.5 threshold is therefore a threshold choice, not a ranking failure. Lowering LR's threshold, the threshold-adjustment technique in module 2.8.5, would recover recall at some cost to precision.

6. Conclusion

On this dataset, Multinomial Naive Bayes is the better spam filter as measured by F1 (0.939 at 70/30, 0.929 at 90/10), because it catches roughly 91% of spam while keeping false positives low. Logistic Regression is the safer choice when a false positive is costly, since it produced at most one across both splits, but it misses 17 to 21% of spam at the default threshold. Increasing the training share from 70% to 90% helped Logistic Regression modestly and Naive Bayes not at all, and the small 90/10 test set makes those differences hard to trust. The larger lesson is that on imbalanced data the metric chosen matters more than the split: accuracy would have called these four models equivalent, and recall shows they are not.

There are two follow-up experiments worth running. (1) Class weighting or threshold tuning (module 2.8.5) would likely close LR's recall gap without retraining. (2) Replacing raw counts with TF-IDF weighting would down-weight common words and may help both models.